Testando smart contracts utilizando Truffle na rede RSK

2020-07-01T19:00:00.000Z

Title

Title

Neste tutorial eu mostrarei passo-a-passo como realizar testes em smart contracts utilizando Truffle framework conectado a um nó local da Blockchain RSK.

# Overview

Aqui está um resumo das etapas que faremos para criar nosso projeto:

1. Instalar pré requisitos;
2. Executar um nó local da RSK (regtest);
3. Criar o projeto Register;
4. Configurar o Truffle;
5. Criar o projeto Register;
6. Criar um smart contract;
7. Compilar;
8. Fazer testes sem deploy;
9. Publicar na rede;
10. Fazer testes após deploy;

# Pré-requisitos

1. POSIX compliant shell
2. Java
3. Node.js e NPM (Node Package Manager)
4. Editor: Visual Studio Code (VSCode) ou outro editor de sua escolha
5. Truffle

Todos os pré-requisitos são explicados detalhadamente no tutorial:

* [Como criar um projeto utilizando Truffle e OpenZeppelin conectado à rede RSK](https://solange.dev/2020/2020-05-10-Rsk-SetupTruffleOZ/)

# Nó local RSK - regtest

Quando desenvolvemos um projeto utilizando o framework Truffle, precisamos de um blockchain executando localmente. Esta é a melhor forma para desenvolver os projetos e executar testes. Nós vamos executar um nó local da rede RSK, também conhecido como regtest.

Existem diversas maneiras para instalar / configurar um nó RSK.

Para aprender como fazer o download de um arquivo JAR e executar com o Java SDK já instalado, veja o tutorial:

* [Como criar um projeto utilizando Truffle e OpenZeppelin conectado à rede RSK](https://solange.dev/2020/2020-05-10-Rsk-SetupTruffleOZ/)

## Execução

Para executar o nó:

java -cp <PATH-TO-THE-RSKJ-JAR> -Drpc.providers.web.cors=\* co.rsk.Start --regtest

(Altere para a localização do seu arquivo JAR).

Eu estou utilizando o sistema operacional (SO) Windows e salvei o arquivo JAR em C:\RSK\node, portanto o caminho completo do meu arquivo é C:\RSK\node\rskj-core-1.3.0-WASABI-all.jar.

Então, para executar o nó RSK:

#### Windows

java -cp C:\RSK\node\rskj-core-2.0.1-PAPYRUS-all.jar -Drpc.providers.web.cors=\* co.rsk.Start --regtest

#### Usando Git Bash

java -cp C:/RSK/node/rskj-core-2.0.1-PAPYRUS-all.jar -Drpc.providers.web.cors=\* co.rsk.Start --regtest

#### Linux ou Mac

java -cp ~/RSK/node/rskj-core-2.0.1-PAPYRUS-all.jar -Drpc.providers.web.cors=\* co.rsk.Start --regtest

Se não apareceu nada depois que você executou o comando, normalmente isso quer dizer que o nó está funcionando perfeitamente. O resultado está sendo salvo no arquivo de log.

**Importante:**

Não feche o terminal / janela do console. O nó está sendo executado nesta janela e, se você fechar, vai encerrar a execução.

# Crie o projeto Register

1. Crie uma nova pasta chamada Register
2. Inicialize um novo projeto Truffle;
3. Inicialize um projeto npm;

São estes os comandos:

mkdir Register  
cd Register  
truffle init  
npm init -y

Por exemplo, eu vou criar a pasta neste local: C:\RSK\

Então meu projeto pode ser localizado no diretório: C:\RSK\Register.

Register project

Register project

Register project

Register project

Register project

Register project

Se quiser mais detalhes sobre esta parte, veja o tutorial já citado anteriormente: [Como criar um projeto utilizando Truffle e OpenZeppelin conectado à rede RSK](https://solange.dev/2020/2020-05-10-Rsk-SetupTruffleOZ/)

Abra o projeto no VS Code. No Windows, isto pode ser feito no terminal:

code .

# Configure o Truffle

Só precisaremos conectar o Truffle ao nó local RSK, então o arquivo truffle-config.js será mais simples. No diretório do projeto, Register, abra o arquivo no VS Code e sobrescreva como o seguinte código:

module.exports = {  
 networks: {  
 development: {  
 host: "127.0.0.1",  
 port: 4444,  
 network\_id: "\*"  
 },  
 },  
 compilers: {  
 solc: {  
 version: "0.5.16",  
 }  
 }  
}

Confira na imagem do VS Code:

truffle-config

truffle-config

# Smart contract

No terminal, no diretório do projeto, execute o comando:

truffle create contract Register

Este comando cria um ‘esqueleto’ de smart contract. Veja o resultado na pasta contracts:

truffle create contract Register

truffle create contract Register

Substitua o conteúdo por este:

pragma solidity 0.5.16;  
  
contract Register {  
 string private info;  
  
 function setInfo(string memory \_info) public {  
 info = \_info;  
 }  
  
 function getInfo() public view returns (string memory) {  
 return info;  
 }  
}

Esta é a imagem no Visual Studio Code:

VS Code Register.sol

VS Code Register.sol

## register.sol

Este smart contract tem:

* A variable info, privada, para armazenar uma string
* A function getInfo() para retornar a string armazenada na variável info
* A function setInfo() para alterar a string armazenada na variável info

# Compile o smart contract

No terminal, execute este comando:

truffle compile

truffle compile

truffle compile

# Testes sem Deploy

É possível testar um smart contract ainda não publicado em nenhuma rede.

Na pasta test, crie o arquivo register\_new.js

Copie e cole este código fonte:

const Register = artifacts.require('Register');  
   
contract('Test new Register', (accounts) => {  
 it('should store an information', async () => {  
 const RegisterInstance = await Register.new();  
 // Set information "RSK"  
 await RegisterInstance.setInfo("RSK", { from: accounts[0] });  
 // Get information value  
 const storedData = await RegisterInstance.getInfo();  
 assert.equal(storedData, "RSK", 'The information RSK was not stored.');  
 });  
});

Veja no VS Code:

VSCode register\_new

VSCode register\_new

Para fazer testes em um smart contract ainda não publicado, utilize o .new(). Isto vai criar uma nova instância para os testes.

## Execute os testes

No terminal, execute este comando:

truffle test

Veja o resultado:

truffle test

truffle test

Nosso teste passou :)

# Publique o smart contract na rede local

Primeiro é necessário criar um arquivo na estrutura do Truffle com instruções para publicar o smart contract.

## Crie o arquivo para deploy

O diretório migrations contém arquivos JavaScript que auxiliam a publicação de contratos no blockchain. Para saber mais, veja [running migrations](https://www.trufflesuite.com/docs/truffle/getting-started/running-migrations)

Na pasta migrations, crie o arquivo 2\_deploy\_contracts.js

Copie e cole este código:

const Register = artifacts.require("Register");  
  
module.exports = function(deployer) {  
 deployer.deploy(Register);  
};

Veja no Visual Studio Code:

2\_deploy\_contracts.js

2\_deploy\_contracts.js

## Migrate

No terminal, execute este comando:

truffle migrate

Como a configuração default do truffle-config.js é para um nó local, leva apenas alguns segundos para que as transações da publicação do smart contract sejam incluídas no Blockchain.

O comando migrate pode compilar o smart contract de novo, se for necessário.

truffle migrate compile

truffle migrate compile

Primeiro é publicado o smart contract Migrations.sol, arquivo gerado pelo Truffle:

truffle migrate Migrations.sol

truffle migrate Migrations.sol

Em seguida, é publicado nosso smart contract register.sol:

truffle migrate Register.sol

truffle migrate Register.sol

# Testes após Deploy

Vamos criar um novo arquivo de testes para o smart contract publicado.

No terminal, no diretório do projeto, execute o comando:

truffle create test Register

Este comando criará um arquivo de testes, com um teste que verifica se o smart contract publicado pode ser instanciado.

Veja no VS Code:

truffle create test Register

truffle create test Register

Vamos alterá-lo para executar os testes que já fizemos antes, agora para o smart contract deployed. Substitua o conteúdo do arquivo por este:

const Register = artifacts.require('Register');  
   
contract('Test deployed Register', (accounts) => {  
 it('should store an information', async () => {  
 const RegisterInstance = await Register.deployed();  
 // Set information "RSK"  
 await RegisterInstance.setInfo("RSK", { from: accounts[0] });  
 // Get information value  
 const storedData = await RegisterInstance.getInfo();  
 assert.equal(storedData, "RSK", 'The information RSK was not stored.');  
 });  
});

Test deployed Register

Test deployed Register

## Execute o teste

O comando truffle test executa os testes de todos os arquivos encontrados no diretório test. Agora temos 2 arquivos com testes. Para executar os testes apenas de um arquivo, especifique o nome no diretório test.

Vamos executar os testes do arquivo register.js. No terminal, execute este comando:

truffle test test/register.js

Veja o resultado:

truffle test register.js

truffle test register.js

Nosso teste novamente passou :)

# Considerações finais

Neste tutorial eu mostrei como utilizar o Truffle framework para realizar testes em smart contracts, conectado a um nó local da Blockchain RSK.

Nosso objetivo é unir forças e dar opções para as pessoas que acreditam poder dos smart contracts baseados em Ethereum e também na segurança do Bitcoin, fazer isso na rede RSK.

Espero que esse tutorial tenha sido útil e agradeço caso tenha algum feedback para mim. Compartilhe o artigo caso tenha gostado :)

Se quiser aprender com videos, assine meu canal: youtube Solange Gueiros